Stork Data Scheduler: Current Status and Future Directions

Sivakumar Kulasekaran

Center for Computation & Technology
Louisiana State University

April 15, 2010
Roadmap

✧ Stork – Data aware Scheduler
✧ Current Status and Features
✧ Future Plans
✧ Application Areas
Motivation

- In a widely distributed computing environment:
  - data transfer performance between nodes may be a major performance bottleneck

- High-speed networks are available, but users may only get a fraction of theoretical speeds due to:
  - unscheduled transfer tasks
  - suboptimal protocol tuning
  - mismanaged storage resources
Data-Aware Schedulers Stork

- Type of a job?
  - transfer, allocate, release, locate..
- Priority, order?
- Protocol to use?
- Available storage space?
- Best concurrency level?
- Reasons for failure?
- Best network parameters?
  - tcp buffer size
  - I/O block size
  - # of parallel streams
Data-aware Scheduling

- Transfer $k$ files between $m$ sources and $n$ destinations, optimize by:
  - Choosing the best transfer protocol; translations between protocols
  - Tuning protocol transfer parameters (considering current network conditions)
  - Ordering requests (considering priority, file size, disk size etc.)
  - Throttling - deciding number of concurrent transfers (considering server performance, network capacity, storage space, etc.)
  - Connection & data aggregation
More Stork features

✧ Queuing, scheduling and optimization of transfers
✧ Plug-in support for any transfer protocol
✧ Recursive directory transfers
✧ Support for wildcards
✧ Checkpointing transfers
✧ Check-sum calculation
✧ Throttling
✧ Interaction with workflow managers and high level planners
Features of Stork 1.2

✧ Current release Stork Version 1.2
✧ Almost available in 17 different platforms
✧ Source code and binary forms of release
✧ Two types of release
  ✧ Core Stork modules
  ✧ Stork with all external modules
Features of Stork 1.2

- First Stand alone version of Stork
- Easy installation steps than previous versions
- Support team to answer all your questions and to provide required help on Stork
- Flexibility for users to customize stork and implement new features
- Test suites to test the functionality of Stork
- Newly updated user friendly Stork user manual
Externals Supported By Stork

- GLOBUS
- OpenSSL
- SRB
- iRods
- Petashare
Optimization Service

✧ To increase wide area throughput by using multiple parallel streams
✧ Opening too many streams results in bottleneck
✧ Important to decide on the optimal number of streams
✧ Predicting optimal number of streams is not easy
✧ Next release of Stork will include optimization features provided by Yildirim et al\(^1\)

Optimization Service
End-to-end Problem

• In a typical system, the end-to-end throughput depends on the following factors:
End-to-end Optimization

• To optimize the total throughput $T_{\text{opt}}$, each term must be optimized

$$T_{\text{opt}} = \min\{\text{opt}\{T_{S_{\text{disk \to mem}}}\}, \text{opt}\{T_{S_{\text{mem \to Network}}\}, \text{opt}\{T_{\text{Network}}\}, \text{opt}\{T_{D_{\text{Network \to mem}}\}, \text{opt}\{T_{D_{\text{mem \to disk}}\}}\}$$
Data Flow Parallelism

Parameters to be optimized
- # of disk stripes
- # of CPUs/nodes
- # of streams
- buffer size per stream
Application Areas

- Coastal & Environment Modeling (SCOOP)
- Reservoir Uncertainty Analysis (UCoMS)
- Computational Fluid Dynamics (CFD)
- Bioinformatics (ANSC)
Other Groups

- CyberTools
- LONI Institute
- MIT
- University of Calgary, Canada
- Offis Institute for Informatics, Germany
- Illuminate Labs
Future Directions

1. Windows Portability

2. Distributed Data Scheduling
   - Interaction between data scheduler
   - Better parameter tuning and reordering of data placement jobs
   - Job Delegation
   - peer-to-peer data movement
Questions

Team

Tevfik Kosar  kosar@cct.lsu.edu
Sivakumar Kulasekaran sivakumar@cct.lsu.edu
Brandon Ross  bross@cct.lsu.edu
Dengpan Yin dyin@cct.lsu.edu

WWW.STORKPROJECT.ORG